GB 2291990A

(12) UK Patent Application (19) GB (11) 2 291 990 (13) A

(43) Date of A Publication 07.02.1996

(21) Application No 9519669.7

(22) Date of Filing 27.09.1995

(71) Applicant(s)
Memory Corporation

(Incorporated in the United Kingdom)

The Computer House, Dalkeith Palace, Dalkeith, EDINBURGH, EH22 2NA, United Kingdom

(72) Inventor(s)
Alan Welsh Sinclair

(74) Agent and/or Address for Service
Rory Macleod
The Computer House, Dalkeith Palace, Dalkeith, EDINBURGH, EH22 2NA, United Kingdom

(51) INT CL[6]
G06F 12/10

(52) UK CL (Edition O)
G4A ANV

(56) Documents Cited
EP 0263014 A1
Dialog record 01599580 of EXE, v7, n10, p72(4), Collinson P, "File systems"

(58) Field of Search
UK CL (Edition N) G4A AND ANV
INT CL[6] G06F 12/02 12/06 12/10, G11C 8/00
On-line : WPI, INSPEC, COMPUTER

(54) Flash-memory management system

(57) A method of maintaining a mapping between a logical sector address and a corresponding physical address using a data storage block for each logical sector address value, and a plurality of spare data storage blocks. Each data storage block comprises: a direct address field 14 and an indirect address field 16. The first time data is written to the logical sector address the direct address field is programmed with the corresponding physical address. Each subsequent time that data is written to the same logical sector address the indirect address field is programmed with the address of a spare data storage block 22 and the direct address field of the spare data storage block is programmed with the new physical address of the data. The physical address accesses a memory 20. The storage blocks are cleaned up when the number of unused blocks drops below a predetermined threshold. The invention allows data updating on non-volatile block-erasable memory such as Flash Eprom. A flag bit 18 may be used to indicate whether the direct or indirect address is valid for accessing memory 20.

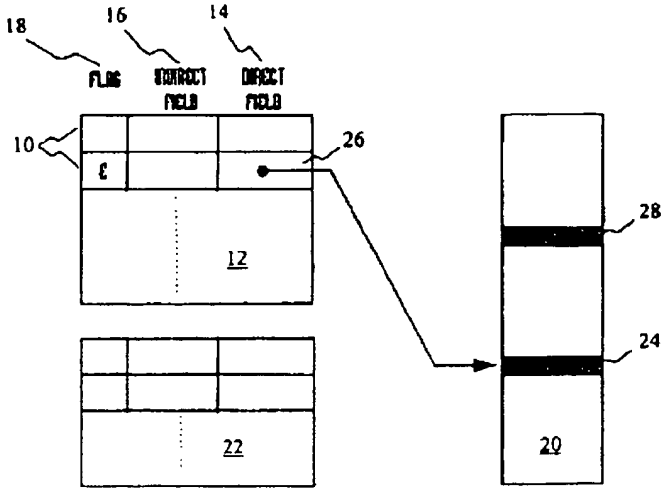

FIGURE 1A

GB 2 291 990 A

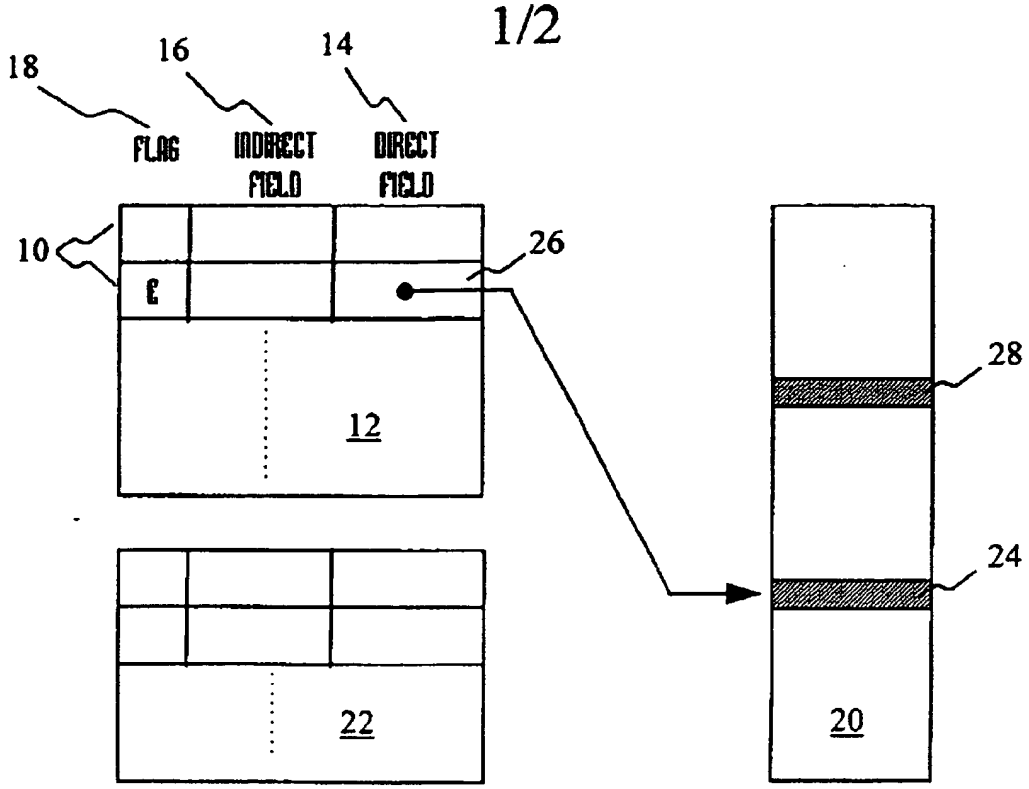

FIGURE 1A

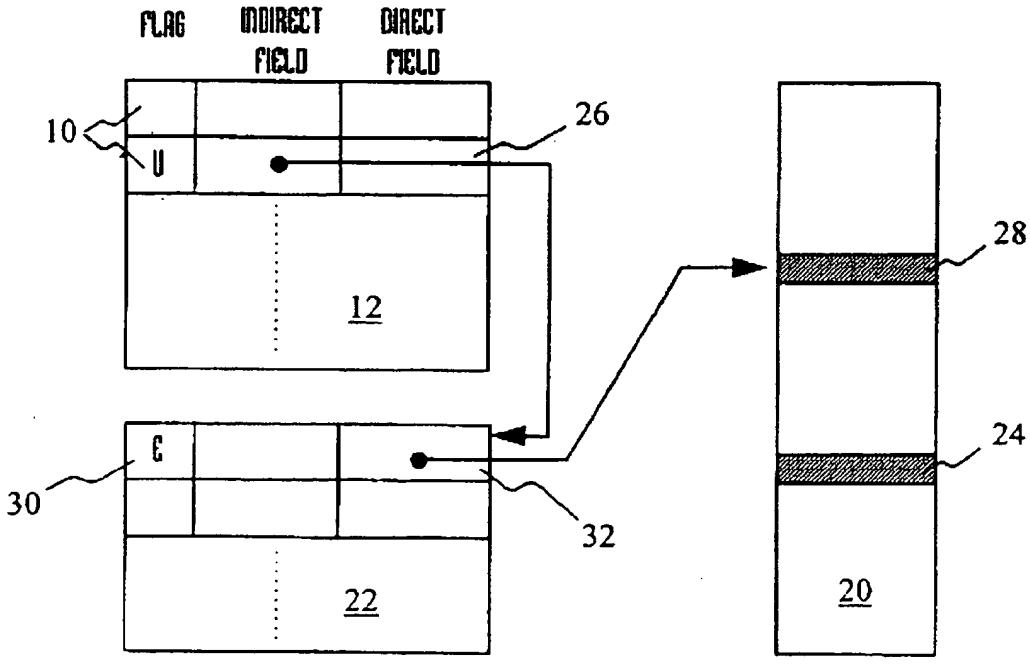

FIGURE 1B

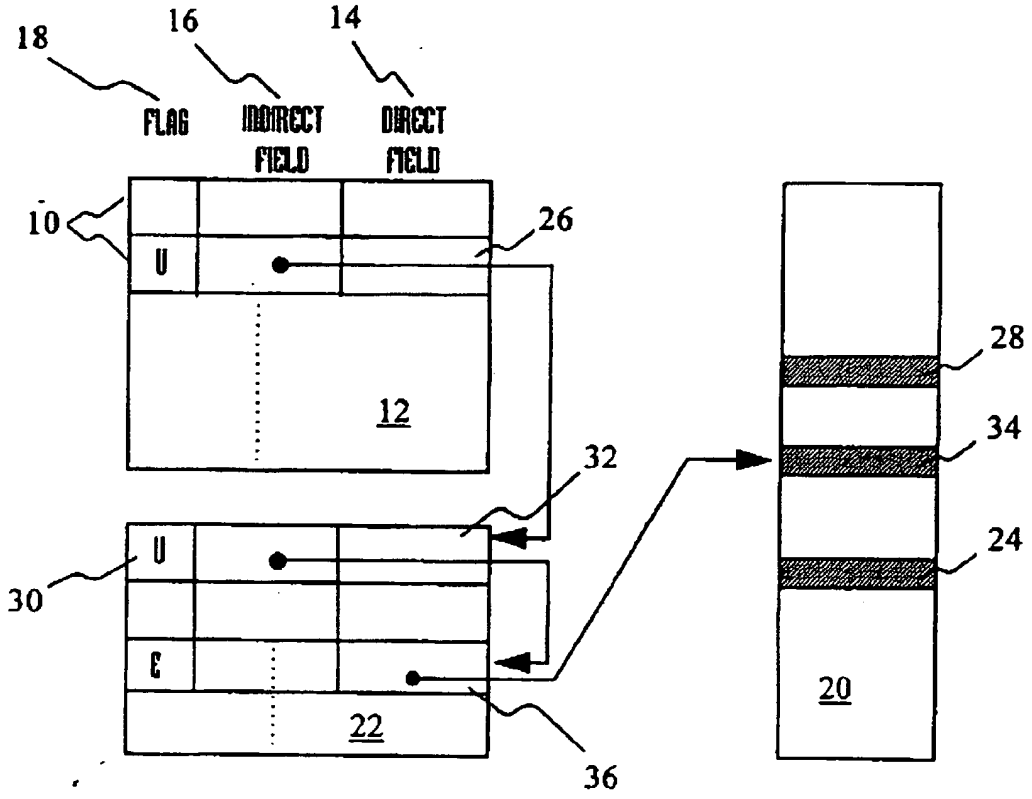
18
16
14
FLAG
INDIRECT FIELD
DIRECT FIELD
10
U
26
12
28
34
32
U
30
24
C
20
22
36

FIGURE 1C

FLASH Memory Management System

The present invention relates to a method of managing a memory in which data can only be written in unwritten memory locations, so that the information stored in the memory can be updated without erasing entire blocks each time the data stored is to be changed

One application of the present invention is in solid state memories which use FLASH memory for the main storage of data and which use FLASH lookup tables to perform the logical to physical address translation. This invention could be used with a number of different memory cells, such as Flash EPROM cells, chalcogenide memory cells, and ferro-optic cells.

Solid state disks receive addresses from the host they are serving in the form of a logical sector address (or by a transformation of Cylinder-Head-Sector addressing from the host). This logical sector addressing is continuous. The logical sector address is mapped to a physical sector address. The physical sector address defines the ordering of sectors in physical memory. Since logical sectors may be assigned to physical sectors in any order, the translation cannot be performed by an algorithm: some form of mapping is required. This mapping is usually performed by means of a lookup table. However, if the lookup table is also composed of FLASH memory cells then individual entries can only be updated by means of a block erasure. This is very inefficient.

In recent years there has been a flurry of activity in the realm of solid state disk devices. This activity has been intensified with the advent of laptop and palmtop portable computers, and the PCMCIA standard. A number of patent applications have been filed which relate to the control methods needed for using FLASH memory as the main data store.

WO 94 20906 relates to a FLASH file system. It uses a block allocation map to store information on the blocks in main FLASH memory which can be written to. When a block is to be updated the allocation map is scanned until a free block is located. A transfer unit is used to facilitate memory reclaiming of the system. The transfer unit is an unwritten erased block of memory. The active (currently valid) data from a block of memory that contains old data (the old block) is written to the same locations in the transfer unit (the new block) and the old block is erased. The old block then becomes the transfer unit. The invention disclosed in WO 94 20906 requires a random access memory to reflect the changes in the block allocation map. The present invention does not require a random access memory, all of the functions necessary to manage the data updating can be performed on non-volatile block erasable memory such as FLASH EPROM.

EP 0 597 706 relates to a solid state peripheral storage device which uses a MAP ROM to map the logical sector address provided by the host to a physical address in the semiconductor memory. The addresses of any bad sectors in semiconductor memory are recorded in the MAP ROM. A microsequencer is used to control the mapping. The MAP ROM is updated when a new physical address is required. The invention relates to floating-gate memories; the method of updating the MAP ROM described in the patent application shows that the MAP ROM could not be a FLASH memory because the MAP ROM must be byte erasable to enable each entry to be updated independently. In contrast, the present invention uses FLASH memory to perform a function equivalent to the MAP ROM function of EP 0 597 706. The advantages of using FLASH memory include: it is more dense than EEPROM, it is cheaper than EEPROM, and it is being used in the solid state disk already.

WO 95 10083 discloses the use of a CAM (content addressable memory) to search for an address which includes a flag field. The address is searched so that only the address with its flag set to a particular value is found as a match. When an address stores data which becomes obsolete its flag is reset. Thus, only the address with valid data has its flag set, the same logical address (but different physical address) having old data has its flag reset.

One advantage of the present invention is that it does not need random access memory such as SRAM or DRAM to map the relation between logical block addresses and physical block addresses.

The present invention provides a method of maintaining a mapping between a first address and a corresponding second address using a data block for each said first address value, with a plurality of spare data blocks, where each data block comprises: a direct address field and an indirect address field; characterised in that when data is initially written to the said first address the direct address field is programmed with the corresponding second address, and each subsequent time that data is written to the said first address the indirect address field is programmed with the address of a spare data block and the direct address field of the said spare data block is programmed with a new second address corresponding to the said first address.

A method of maintaining a mapping between a first address and a corresponding second address where the said data block includes a pointer flag.

A method of maintaining a mapping between a first address and a corresponding second address where the said pointer flag can be written to independently of the rest of the said data

block.

A method of maintaining a mapping between a first address and a corresponding second address where the said pointer flag is in the erased state until the indirect address field of the corresponding data block is programmed.

A method of maintaining a mapping between a first address and a corresponding second address wherein the pointer flag is read to determine whether the direct address field or the indirect address field is to be read.

A method of maintaining a mapping between a first address and a corresponding second address where the data blocks are cleaned-up when a predetermined condition is reached.

A method of maintaining a mapping between a first address and a corresponding second address where the data blocks are cleaned-up when the number of unused data blocks drops below a predetermined threshold.

Figures 1A, 1B and 1C illustrate the principle of translating between two addresses using an indirect address field and a direct address field.

Every solid state memory system must convert logical sector addresses received from the host to physical addresses which can be used to address the particular solid state memory devices which are used as the main memory store. Thus, some logical to physical address conversion is required. This embodiment of the present invention uses two lookup tables to maintain a mapping between a logical sector address and its corresponding physical sector address. In other embodiments of this invention only one table may be used, where the table performs the same function as the two lookup tables.

Referring to Figure 1A, a lookup table entry 10 for each logical sector address value is located in the primary lookup table 12 at a location defined by an address equal to the logical sector address plus the primary lookup table offset.

Each entry in the primary lookup table 12 contains at least three fields: one field stores a main memory address 14; another field stores a linked address 16 ; and the third field is a pointer flag 18. Each used entry in the primary lookup table 12 either points directly at the physical address of a sector in main memory 20 or provides an indirect pointer to a secondary lookup table 22. In the present embodiment the means of determining whether the primary lookup table entry points to the main memory 20 or whether it points to the secondary lookup table 22 is to read the pointer flag 18 in the relevant table entry. If the pointer flag 18 is erased (shown by the letter E in Figures 1A and 1B) then the main memory address is valid, if the pointer is

unerased (shown by the letter U in Figures 1A and 1B) then the secondary lookup table address is valid.

Consider the case when there is no data stored in the main memory 20. When data is first written then a logical sector address is supplied by the host together with the data to be stored. The data is stored in a physical location 24 in main memory 20 and the address of this physical location 24 in main memory 20 is stored in the primary lookup table 12 at an entry 26 corresponding to the logical sector address supplied by the host.

If, at some later time the data is deleted by the host and new data is sent with the same logical sector address as the old data, then the new data is stored in an unused main memory location 28 (the previous main memory location 24 cannot be overwritten because FLASH memory needs an erase cycle after being written). The primary lookup table 12 must now be updated. This is illustrated in Figure 1B.

Figure 1B illustrates how the lookup tables are updated to ensure correct mapping between the logical sector address and the new physical address.

The entry 26 in the primary lookup table 12 now needs to be updated with the address of the main memory location 28 for the new data. This is done by writing the pointer flag in the entry corresponding to the logical sector address 26 to logic zero (unset) and the linked address to one of the unused entries in the secondary lookup table 22 (hereinafter referred to as the xxx secondary table entry 32). The pointer flag can be written because it occupies an addressable unit and because it has not been written to since the erase cycle: the inactive state of the pointer is the erased state (logic one). The address of the memory location to which the new data was written is stored in the main memory address field 14 of the xxx secondary lookup table entry 32.

If, at some later time, the new data is deleted by the host and more data is sent (this will be referred to as the latest data) then this latest data is stored in another unused location (this will be referred to as the latest unused location) in main memory 34. This is illustrated in Figure 1C. The primary lookup table 12 is not altered, but the xxx entry in the secondary lookup table 32 now has its pointer flag 30 written so that it is unset (logic zero) and the linked address field is written so that it points to an unused entry in the secondary lookup table (this will be referred to as the yyy secondary lookup table entry 36). The address of the latest unused location is then stored in the main memory field 14 of the yyy secondary lookup table entry 36. This process may continue until the number of unused entries in the secondary lookup table 22

drops below a predetermined value.

Once the secondary lookup table 22 becomes full then a clean-up operation is required. When the primary lookup table 12 is cleaned-up, a new lookup table must be created before the old one is erased to safeguard the information. One way of performing a clean-up operation is to reserve at least one erasable FLASH block in the erased state either above or below the primary lookup table 12. One entry from the primary lookup table 12 is copied to the erased block and the old entry is then erased. As each entry is copied, the main memory address field 14 for that entry is copied from the relevant valid secondary lookup table entry (the entry with its pointer flag in the erased condition). In effect the primary lookup table 12 moves up or down by one erase block. Once this process is complete the secondary lookup table 22 can be erased.

In other embodiments of the present invention only two fields are used: a main memory address and a secondary lookup table address. If the secondary lookup table address contains valid data then that address is read instead of the main memory address. If the secondary lookup table address does not contain valid data then the main memory address is read instead.

Claims

1. A method of maintaining a mapping between a first address and a corresponding second address using a data block for each said first address value, with a plurality of spare data blocks, where each data block comprises: a direct address field and an indirect address field; characterised in that when data is initially written to the said first address the direct address field is programmed with the corresponding second address, and each subsequent time that data is written to the said first address the indirect address field is programmed with the address of a spare data block and the direct address field of the said spare data block is programmed with a new second address corresponding to the said first address.
2. A method of maintaining a mapping between a first address and a corresponding second address according to claim 1 where the said data block includes a pointer flag.
3. A method of maintaining a mapping between a first address and a corresponding second address according to claim 2 where the said pointer flag can be written to independently of the rest of the said data block.
4. A method of maintaining a mapping between a first address and a corresponding second address according to either of claims 2 or 3 where the said pointer flag is in the erased state until the indirect address field of the corresponding data block is programmed.
5. A method of maintaining a mapping between a first address and a corresponding second address according to any of claims 2,3 or 4, wherein the pointer flag is read to determine whether the direct address field or the indirect address field is to be read.
6. A method of maintaining a mapping between a first address and a corresponding second address according to any preceding claim where the data blocks are cleaned-up when a predetermined condition is reached.
7. A method of maintaining a mapping between a first address and a corresponding second address according to any preceding claim where the data blocks are cleaned-up when the number of unused data blocks drops below a predetermined threshold.

The Patent Office

Application No:	GB 9519669.7	**Examiner:**	B.G.Western
Claims searched:	1-7	**Date of search:**	8 November 1995

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.N): G4A AND ANV

Int Cl (Ed.6): G06F 12/02 12/06 12/10
G11C 8/00

Other: On-line : WPI, INSPEC, COMPUTER

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	EP-0263014-A1 PICARD (N.b Figure 7)	-
A	Dialog record 01599580 of EXE, v7, n10, p72(4), Collinson P, "File systems"	-

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.